Name Of the Project

EMAIL SPAM CLASSIFIER

Submitted By

Mrs. Swati Amit Motugade

FlipRobo SME

Gulshana Chaudhari

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysis skills. Also, I want to express my huge gratitude to Ms. Gulshana Chaudhari Mam (SME, Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project.

A huge thanks to “Data trained” who are the reason behind Internship at Fliprobo. Last but not least my parents who are there to support me at every step of my life.

References used in this project:

1. SCIKIT Learn Library Documentation.
2. Blogs from towardsdatascience, Analytics Vidya, Medium.
3. Andrew Ng Notes on Machine Learning (GitHub).
4. Data Science Projects with Python Second Edition by Packt
5. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
6. https://www.javatpoint.com/nlp

7.https://www.educative.io/answers/preprocessing-steps-in-naturallanguage- processing-nlp

8. https://www.youtube.com/watch?v=5ctbvkAMQO4

9. https://www.youtube.com/watch?v=X2vAabgKiuM

Mrs. Swati Amit Motugade

Chapter 1

Introduction

1.1 Introduction

What is Email?

Email stands for electronic mail. It’s used to send and receive computer stored messages via the internet. Emails are generally encoded in American Standard Code for Information Interchange(ASCII). Generally, an email is a message that may contain text, files, images, or other attachments sent through the computer network to an individual or group of individuals. The email services are far older than the ARPANET and the internet. Email development started in early 1960 when users were able to share messages with other users of the same computer. In 1971, Ray Tomlinson was the first to send mail between two computer systems of ARPANET by adding a program CPYNET he used the ‘@’ sign to link the user and hostname as a part of its addressing system.

Uses of Email:

Email can be used to communicate either within an organization or personally, including between two people or a large group of people. Most people get benefit from communicating by email with colleagues or friends or individuals or small groups. It allows you to communicate with others around the world and send and receive images, documents, links, and other attachments. Additionally, it offers benefit users to communicate with the flexibility on their own schedule.

There is another benefit of using email; if you use it to communicate between two people or small groups that will beneficial to remind participants of approaching due dates and time-sensitive activities and send professional follow-up emails after appointments. Users can also use the email to quickly remind all upcoming events or inform the group of a time change. Furthermore, it can be used by companies or organizations to convey information to large numbers of employees or customers. Mainly, email is used for newsletters, where mailing list subscribers are sent email marketing campaigns directly and promoted content from a company.

Email can also be used to move a latent sale into a completed purchase or turn leads into paying customers. For example, a company may create an email that is used to send emails automatically to online customers who contain products in their shopping cart. This email can help to remind consumers that they have items in their cart and stimulate them to purchase those items before the items run out of stock. Also, emails are used to get reviews by customers after making a purchase. They can survey by including a question to review the quality of service.

Spam Email:

Spam email is unsolicited and unwanted junk email sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. It can be sent in massive volume by botnets, networks of infected computers.

Why Spam email?

Often, spam email is sent for commercial purposes. While some people view it as unethical, many businesses still use spam. The cost per email is incredibly low, and businesses can send out mass quantities consistently. Spam email can also be a malicious attempt to gain access to your computer.

Spam email can be dangerous. It can include malicious links that can infect your computer with malware. Do not click links in spam. Dangerous spam emails often sound urgent, so you feel the need to act. Keep reading to learn about some of the basic spam types.

Common types of Spam email:

There are 7 most common types of spam mails which are explained as below:

1. **Ads:**

This is one of the most common types of spam. I bet you've already received several unsolicited emails offering products and services, such as weight loss pills and tennis offers. In many cases it may be a scam but the offer may also be real.

2. **Chain Letters**

"Something bad will happen to you". Usually, chain letters tell exciting and thrilling stories and persuade you to pass the message along under penalty of having something very bad happen to you. Be careful or you're going to have a run of bad luck. Buuuuuu!

3. **Email spoofing:**

Spoofing email spam is related to phishing scams. They happen when spammers or phishers try to fool you by impersonating someone you know or a company you have a relationship with. This is one of the most dangerous types of spam.

4. **Hoaxes**:

In this type of spam, we include offers and miracle promises, such as, for example, "get rich in less than a month" or "gain the body of your dream by eating more and working out less". In general, this tactic is used by spammers to hold your attention and direct you to a malicious website.

5. **Money Scams:**

Here we've spam messages with easy money promises, such as the Nigerian prince scheme. In this case, you apparently only have to lend a small amount of money to receive a big reward in the future.

Money spam also involves asking for money for hungry children in Africa or for families who have suffered losses as a result of a natural disaster.

6. Malware Warnings:

If you receive an email warning you about a malware infection on your devices, such as ransomware or virus, this is probably a malware warning spam. In some cases, the spammers say they have the solution to your problem and that you just need to provide some information or download an attachment. Watch out!

7. Porn Spam:

We decided to add this type of spam to the list because it's very common. Sending pornography through email is widely used by spammers because the pornography market is very lucrative, increasing people's interest.

1.2 Business Problem Framing

Most of us consider spam emails as one which is annoying and repetitively used for purpose of advertisement and brand promotion. We keep on blocking such email-ids but it is of no use as spam emails are still prevalent. Some major categories of spam emails that are causing great risk to security, such as fraudulent e-mails, identify theft, hacking, viruses, and malware. In order to deal with spam emails, we need to build a robust real-time email spam classifier that can efficiently and correctly flag the incoming mail spam, if it is a spam message or looks like a spam message. The latter will further help to build an Anti-Spam Filter.

Google and other email services are providing utility for flagging email spam but are still in the infancy stage and need regular feedback from the end user. Also, popular email services such as Gmail, Yandex, yahoo mail, etc provide basic services as free to the end-user and that of course comes with EULA. There is a great scope in building email spam classifiers, as the private companies run their own email servers and want them to be more secure because of the confidential data, in such cases email spam classifier solutions can be provided to such companies.

The reason to do this is simple: by detecting unsolicited and unwanted emails, we can prevent spam messages from creeping into the user's inbox, thereby improving user experience.

1.3 Conceptual Background of the Domain Problem

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

This corpus has been collected from free or free for research sources at the Internet:

-> A collection of 5573 rows SMS spam messages was manually extracted from the   Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.

-> A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.

1.4 Review of Literature

Email Spam has become a major problem nowadays, with Rapid growth of internet users, Email spams is also increasing. People are using them for illegal and unethical conducts, phishing and fraud. Sending malicious link through spam emails which can harm our system and can also seek in into your system. Creating a fake profile and email account is much easy for the spammers, they pretend like a genuine person in their spam emails, these spammers target those peoples who are not aware about these frauds. So, it is needed to Identify those spam mails which are fraud, this project will identify those spam by using techniques of machine learning, this paper will discuss the machine learning algorithms and apply all these algorithms on our data sets and best algorithm is selected for the email spam detection having best precision and accuracy.

Despite the advancement of spam filtering applications and services, there is no definitive way to distinguish between legitimate and malicious emails because of the ever-changing content of such emails. Spams have been sent for over three or four decades now, and with the availability of various antispam services, even today, nonexpert end-users get trapped into such hideous pitfall. In e-mail managers, spam filters detect spam and forward it to a dedicated space, spam folder, allowing the user to choose whether or not to access them. Spam filtering tools such as corporate e-mail systems, e-mail filtering gateways, contracted antispam services, and end-user training can deal with spam emails in English or any other language. However, they are ineffective at filtering spam emails in other languages that recently have been digitized.

Precision, recall, and f-measure are considered key evaluating measures to compare Naive Bayes and SVM, while the evaluation parameters, i.e., Model

Loss and ROC-AUC, are calculated for deep learning models such as CNN and LSTM. Finally, a comparison is made between all models for the best accuracy and values of evaluation parameters obtained by DL and ML models

Chapter 2

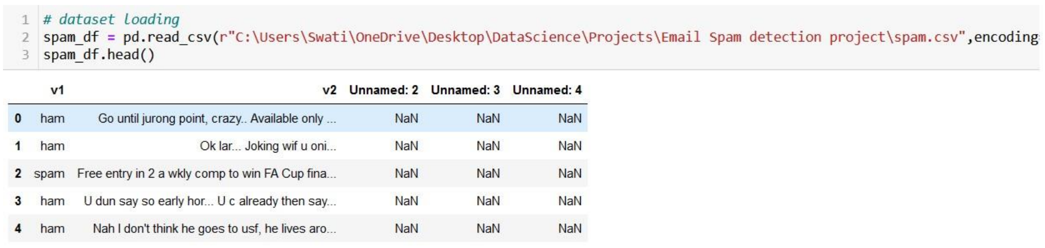
Analytical Problem Framing

2.1 Mathematical Modelling of the Problem

Our objective is to classify the emails as spam or ham using ML algorithms. In this project we are going to use different types of algorithms which uses their own mathematical equation on background. This project comes with a csv data set. Initially data cleaning & pre-processing performed over data. Feature engineering is performed to remove unnecessary feature & for dimensionality reduction. In model building Final model is select based on evaluation benchmark among different models with different algorithms. Further Hyperparameter tuning performed to build more accurate model out of best model.

2.2 Data Sources and their formats

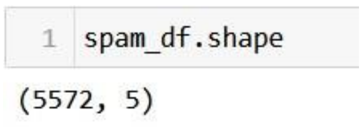
Data set provided by Flip Robo was in the format of CSV (Comma Separated Values).



```
# dataset loading
spam_df = pd.read_csv(r"C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Email Spam detection project\spam.csv",encoding
spam_df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Checking for shape of dataset



```
spam_df.shape
```

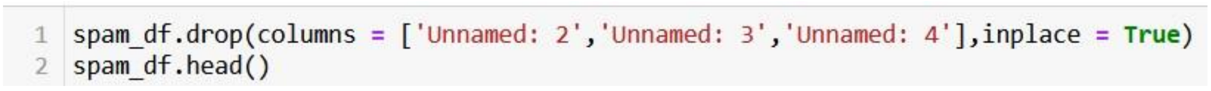
```
(5572, 5)
```

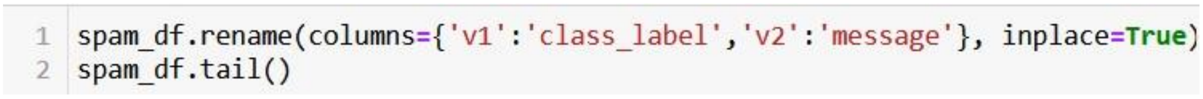
We can see that there are 5572 rows and 5 columns in our dataset.

Out of 5 columns there are 3 columns which contains maximum NaN values. These columns are of no use so we will remove these columns. Out of the remaining columns one is containing label i.e., if the mail spam or ham and the other one contains the actual message.

The column of label is named as v1 and the column of message is named as v2.

For our convenience and better understanding we will rename these columns as ‘class_label’ and ‘message’ respectively.

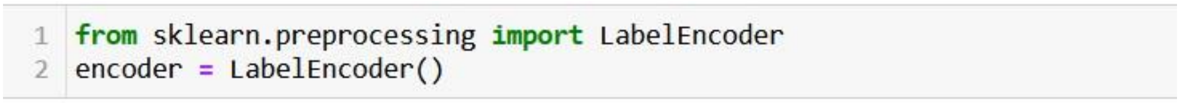

```
spam_df.drop(columns = ['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace = True)
spam_df.head()
```

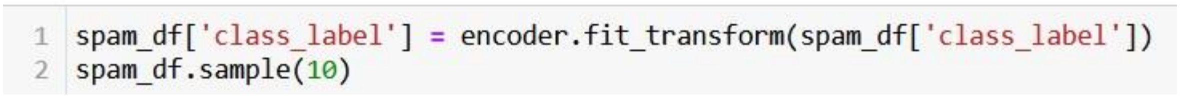

```
spam_df.rename(columns={'v1':'class_label','v2':'message'}, inplace=True)
spam_df.tail()
```

2.3 Data Pre-processing

Label Encoding

Since our class_label column contains labels ‘spam’ and ‘ham’ we have to convert them into numeric form for that we need to perform label encoding.

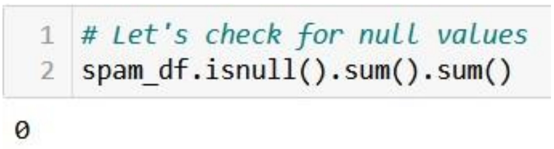

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```


```
spam_df['class_label'] = encoder.fit_transform(spam_df['class_label'])
spam_df.sample(10)
```

After label encoding, we can see that our labels ‘ham’ as 0 and ‘spam’ as 1.

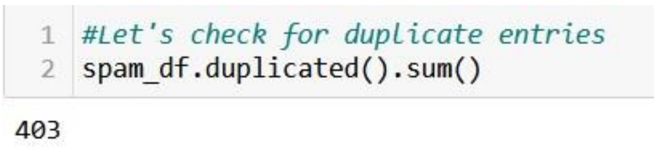
Data Integrity

Here we checked the integrity of our dataset mean we checked for null values and duplicate entries and found that there are no missing values or null values in our dataset but it contains 403 duplicate entries which we dropped.



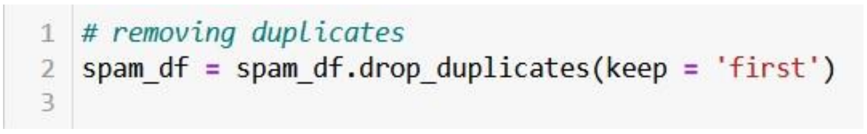
```python
# Let's check for null values
spam_df.isnull().sum().sum()
```

```
0
```

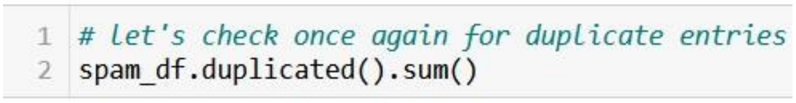


```python
#Let's check for duplicate entries
spam_df.duplicated().sum()
```

```
403
```



```python
# removing duplicates
spam_df = spam_df.drop_duplicates(keep = 'first')

```



```python
# Let's check once again for duplicate entries
spam_df.duplicated().sum()
```

2.4 EDA

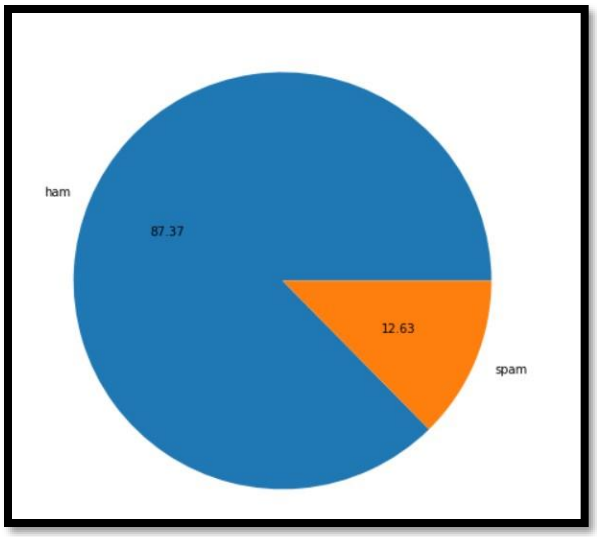
Exploratory Data Analysis (EDA) involves using statistics and visualizations to analyze and identify trends in data sets.

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Since our problem is of classification type, we plotted pie chart to represent our target variable with percentage.

```python
# lets import necessary libraries for plotting
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# plotting a pie chart for class_label feature
plt.figure(figsize=(15,8))
plt.pie(spam_df['class_label'].value_counts(),labels=['ham','spam'],autopct="%.2f")
plt.show()
```



The above pie chart shows that out of all messages, 87.37% messages are ham type and 12.63% of spam type.

Feature Extraction

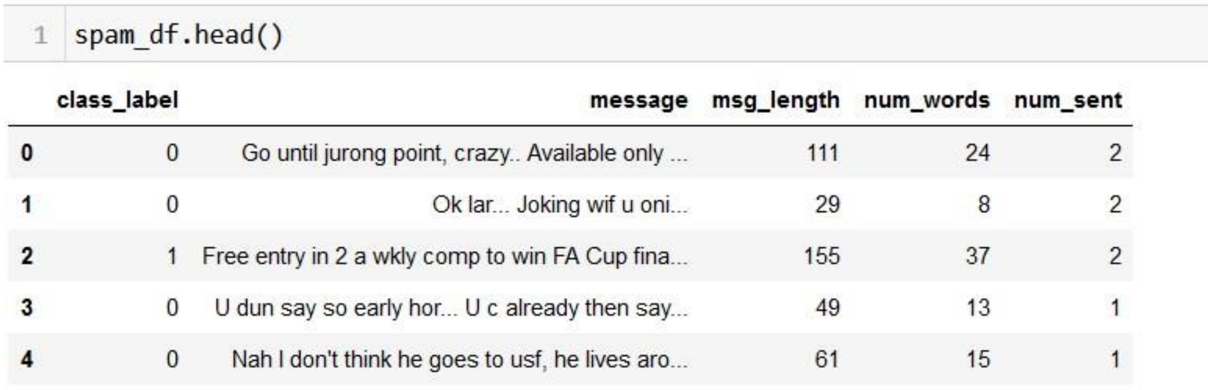
We created 3 new features named as msg_length, num_words and num_sent which represents message length or number of characters, number of words and number of sentences in every message respectively.

```python
# checking for length of each message by applying len function
spam_df['msg_length'] = spam_df['message'].apply(len)
```

```python
# lets check for number of words for every message
spam_df['num_words'] = spam_df['message'].apply(lambda x:len(nltk.word_tokenize(x)))
```

```python
# check for number of sentences
spam_df['num_sent'] = spam_df['message'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

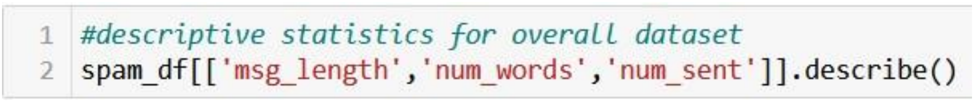
And our dataset with these additional features looks like



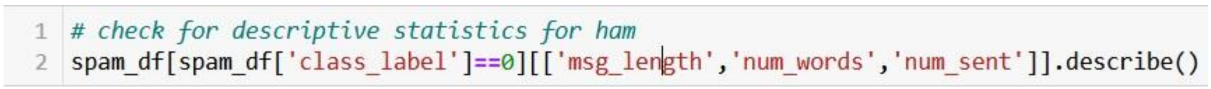
```
spam_df.head()
```

	class_label	message	msg_length	num_words	num_sent
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

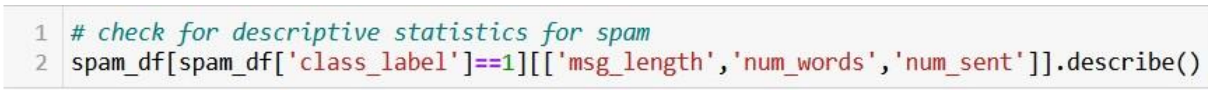
Further we checked for descriptive statistics of the added features using describe function to know about the maximum and minimum number of words and sentences.



```
#descriptive statistics for overall dataset
spam_df[['msg_length','num_words','num_sent']].describe()
```



```
# check for descriptive statistics for ham
spam_df[spam_df['class_label']==0][['msg_length','num_words','num_sent']].describe()
```



```
# check for descriptive statistics for spam
spam_df[spam_df['class_label']==1][['msg_length','num_words','num_sent']].describe()
```

The above descriptive summary gives the result as below:

For the ham messages the maximum length of message is 910 and minimum length is 2 whereas for spam messages the maximum length of messages is 224 and minimum is 13.
For ham messages the maximum number of words is 220 and minimum is 1 whereas for spam messages the maximum number of words is 46 and minimum is 2.

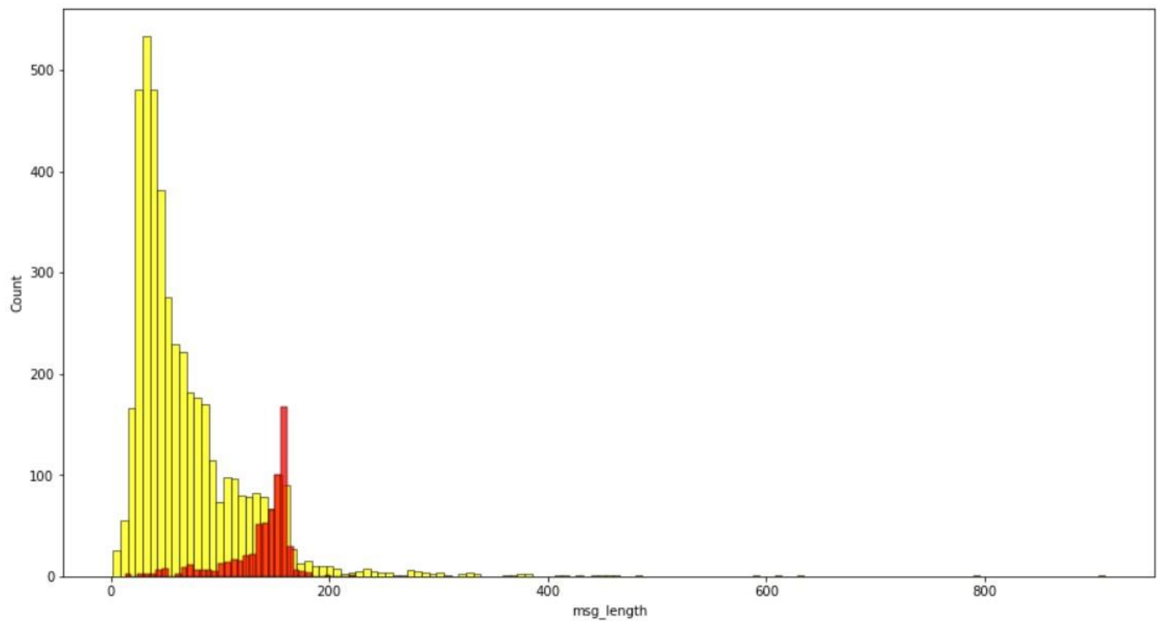
For ham messages the maximum number of sentences is 38 and minimum is 1 whereas for spam messages the maximum number of sentences is 9 and minimum is 1.

Visualization of Newly Added Features

We plotted the histograms for newly added features to know the differences for spam and ham.

```python
# import seaborn to plot histogram
import seaborn as sns
```

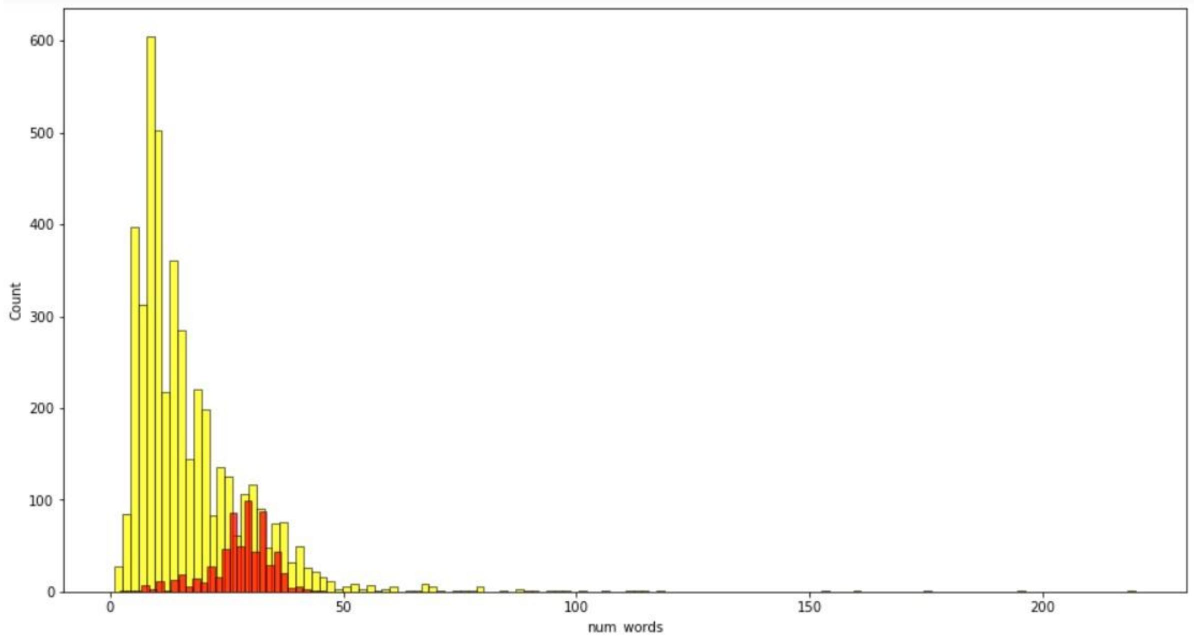
```python
# plotting a histogram for msg_length of spam and ham
plt.figure(figsize=(15,8))
sns.histplot(spam_df[spam_df['class_label']==0]['msg_length'],color='yellow')
sns.histplot(spam_df[spam_df['class_label']==1]['msg_length'],color='red')
plt.show()
```



Here, the yellow lines are representing the length of ham messages and red lines are representing the length of spam messages and we can clearly see that the length of spam messages are very lesser than that of ham messages.

Similarly we plotted the feature num_words.

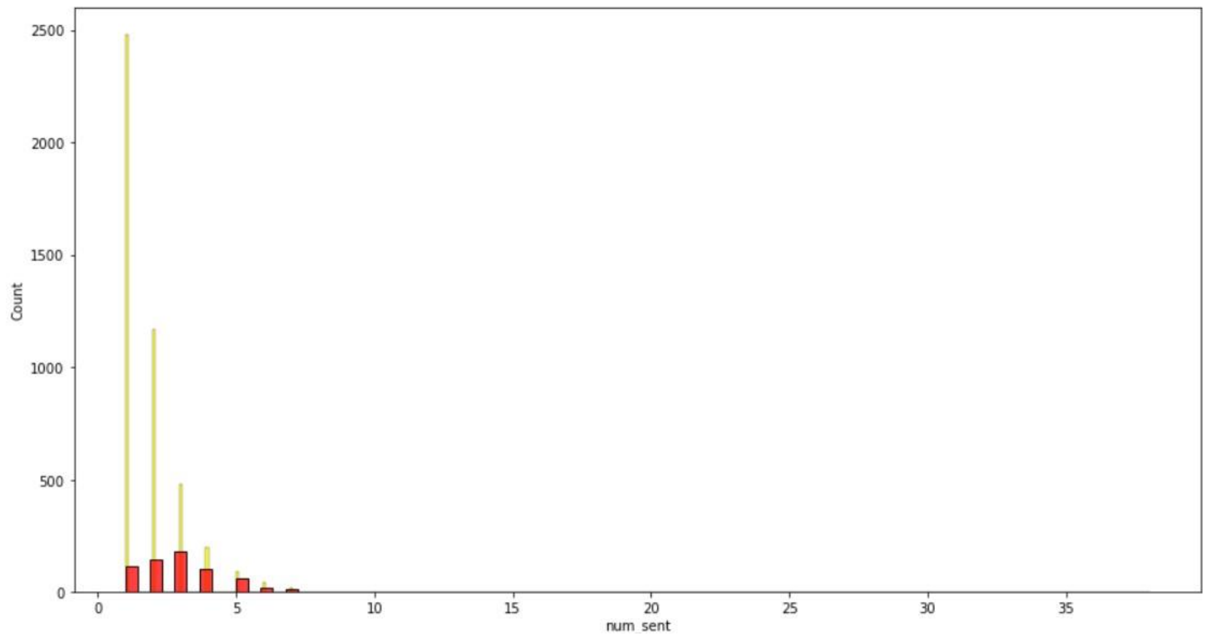
```python
# plotting histogram for num_words in spam and ham
plt.figure(figsize=(15,8))
sns.histplot(spam_df[spam_df['class_label']==0]['num_words'],color='yellow')
sns.histplot(spam_df[spam_df['class_label']==1]['num_words'],color='red')
plt.show()
```



Here also we can see that the number of words in ham messages are more than number of words in spam messages.

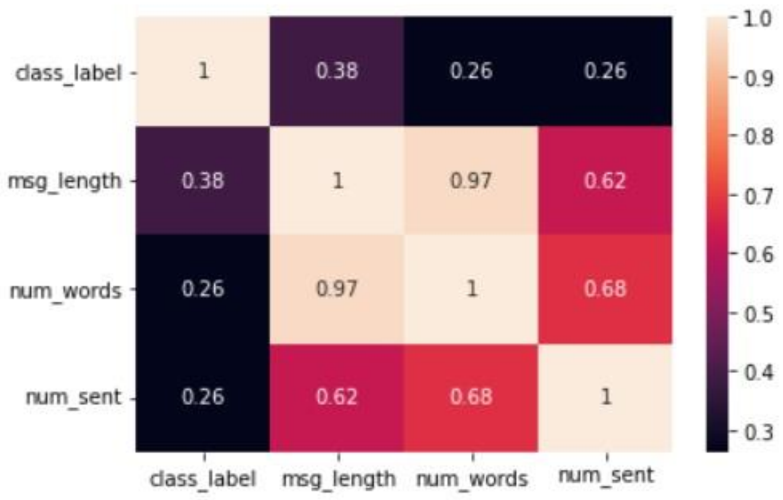
```python
# plotting a histogram for num_sent in spam and ham
plt.figure(figsize=(15,8))
sns.histplot(spam_df[spam_df['class_label']==0]['num_sent'],color='yellow')
sns.histplot(spam_df[spam_df['class_label']==1]['num_sent'],color='red')
plt.show()
```

We



```
# let's plot a heatmap for the correlation function
sns.heatmap(spam_df.corr(),annot=True)
```

```
<AxesSubplot:>
```



Text Normalization

Also, the number of sentences in ham messages are more than that of in spam messages.

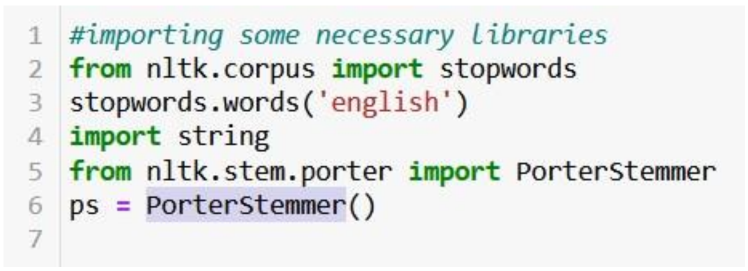
2.4 Correlation

We checked for correlation using corr() function and observed that there is multicollinearity present in between the features msg_length and num_words.

```
#lets check for correlation of dataset
spam_df.corr()
```

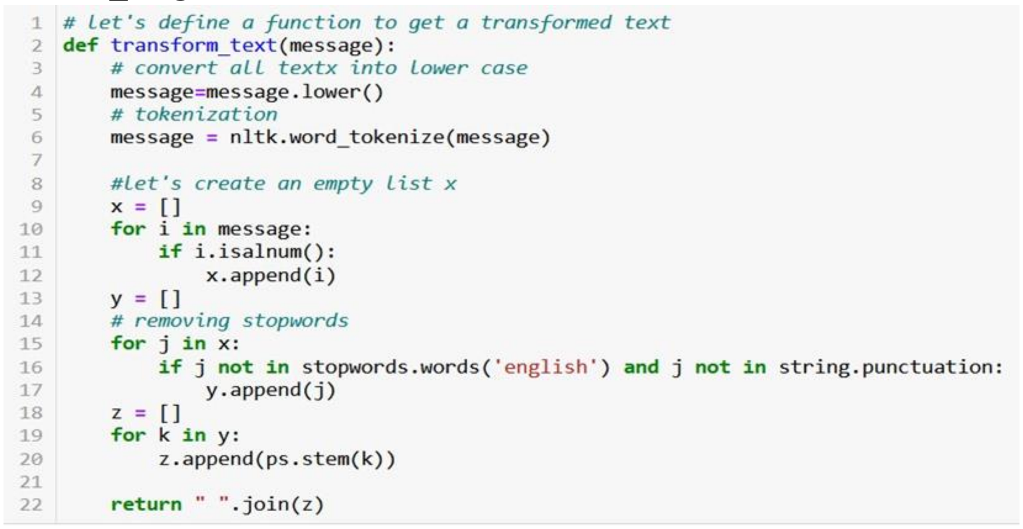
Also we plotted heatmap of correlation function which looks like as below

Stemming and Lemmatization are **Text Normalization** (or sometimes called **Word Normalization**) techniques in the field of **Natural Language Processing** that are used to prepare text, words, and documents for further processing. Stemming and Lemmatization have been studied, and algorithms have been developed in Computer Science since the 1960's. In this tutorial you will learn about Stemming and Lemmatization in a practical approach covering the background, some famous algorithms, applications of Stemming and Lemmatization, and how to stem and lemmatize words, sentences and documents using the Python nltk package which is the Natural Language Tool Kit package provided by Python for Natural Language Processing tasks. Here we imported some libraries for text normalization

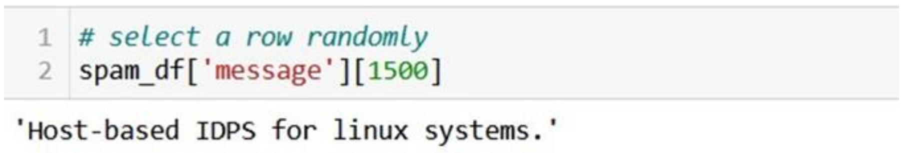


```python
#importing some necessary libraries
from nltk.corpus import stopwords
stopwords.words('english')
import string
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

```

Then defined a function which gives us clean text message and applied that function on the feature ‘message’ to get new feature named as ‘transformed_msg’.

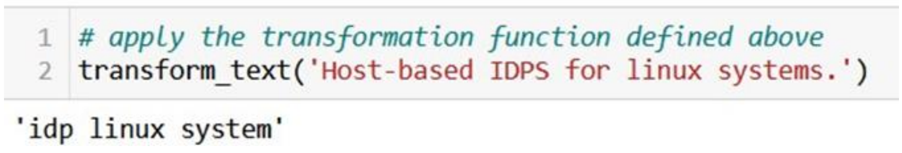


```python
# Let's define a function to get a transformed text
def transform_text(message):
    # convert all textx into lower case
    message=message.lower()
    # tokenization
    message = nltk.word_tokenize(message)

    #let's create an empty list x
    x = []
    for i in message:
        if i.isalnum():
            x.append(i)
    y = []
    # removing stopwords
    for j in x:
        if j not in stopwords.words('english') and j not in string.punctuation:
            y.append(j)
    z = []
    for k in y:
        z.append(ps.stem(k))

    return " ".join(z)
```



```python
# select a row randomly
spam_df['message'][1500]
```

```
'Host-based IDPS for linux systems.'
```



```python
# apply the transformation function defined above
transform_text('Host-based IDPS for linux systems.')
```

```
'idp linux system'
```

```python
# create new column transformed_msg
spam_df['transformed_msg'] = spam_df['message'].apply(transform_text)
```

```python
spam_df.head()
```

	class_label	message	msg_length	num_words	num_sent	transformed_msg
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

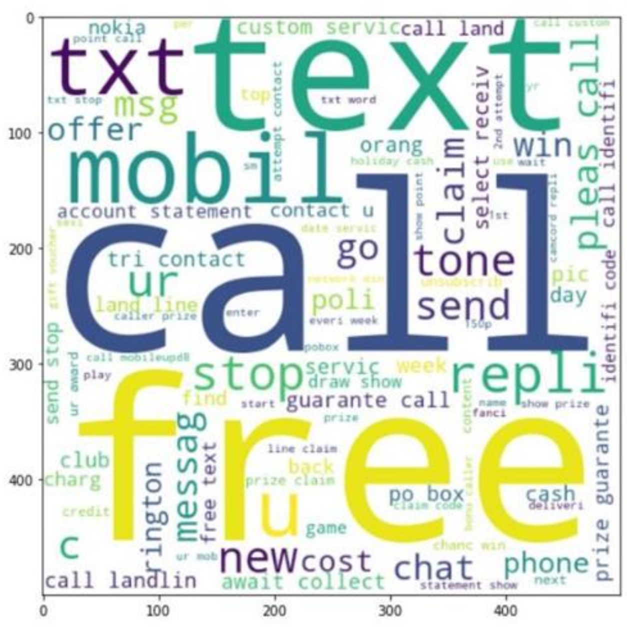
Wordcloud

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites.

plotting wordcloud for spam messages:

```python
# create wordcloud for spam
from wordcloud import WordCloud
wc = WordCloud(width = 500, height = 500, min_font_size = 10, background_color = 'white')
plt.figure(figsize=(12,8))
spam_wc = wc.generate(spam_df[spam_df['class_label']==1]['transformed_msg'].str.cat(sep = ''))
plt.imshow(spam_wc)
```

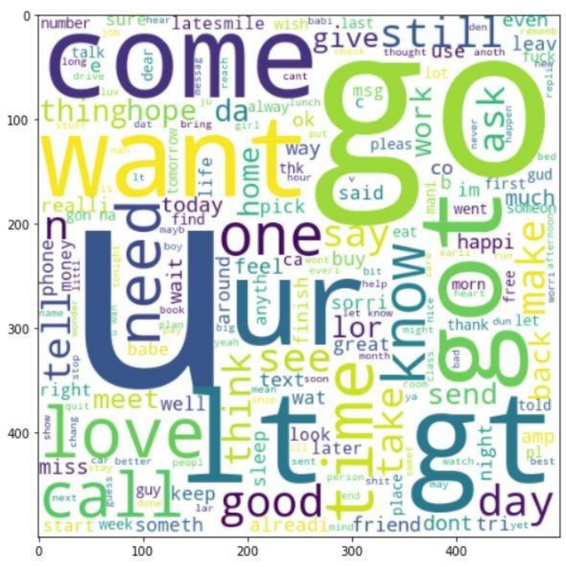
```
<matplotlib.image.AxesImage at 0x245341d1f40>
```



Plotting wordcloud for ham messages

```
# lets create wordcloud for ham
from wordcloud import WordCloud
wc = WordCloud(width = 500, height = 500, min_font_size = 10, background_color = 'white')
plt.figure(figsize=(12,8))
ham_wc = wc.generate(spam_df[spam_df['class_label']==0]['transformed_msg'].str.cat(sep = ''))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x24534d3f520>
```



the words 'call','free','new','mobil','text','txt','offer','repli','claim','stop' etc are the most common words in spam class. And the words 'u', 'ur', 'come', 'go', 'got', 'want', 'day', 'time', 'need', 'know' etc are most common words in ham messages.

Later we checked for 50 most common words in spam and ham messages and made a data frame of those words with their respective frequencies. For spam messages

```
# most common 50 words in spam messages
spam_corpus = []

for msg in spam_df[spam_df['class_label']==1]['transformed_msg'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

```
len(spam_corpus)
```

```
9939
```

```
from collections import Counter
Counter(spam_corpus).most_common(50)
```

```
#lets make a dataframe for most common words of spam messages to show them with their frequencies
pd.DataFrame(Counter(spam_corpus).most_common(50))
```

For ham messages

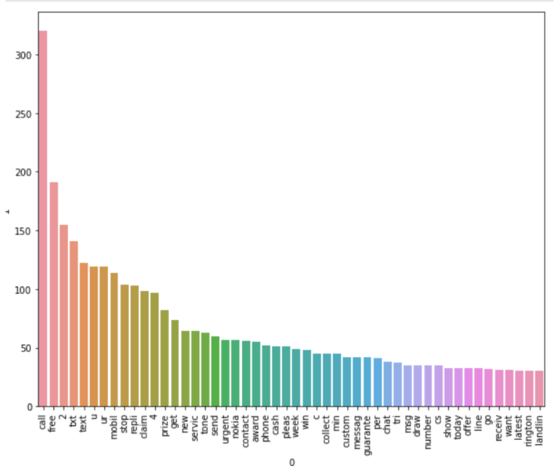
```python
# most common 50 words in ham messages
ham_corpus = []

for msg in spam_df[spam_df['class_label']==0]['transformed_msg'].tolist():
    for word in msg.split():
        ham_corpus.append(word)

```

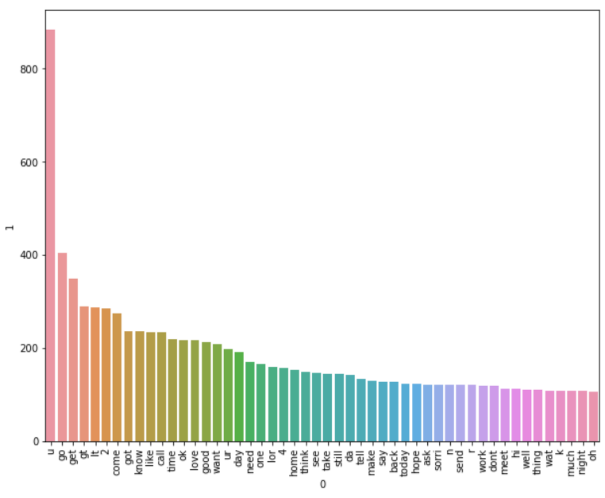
```python
len(ham_corpus)
```

```
35404
```

```python
from collections import Counter
Counter(spam_corpus).most_common(50)
```

Also, we plotted a barplot of these words with their frequencies to get an idea about their occurrences.

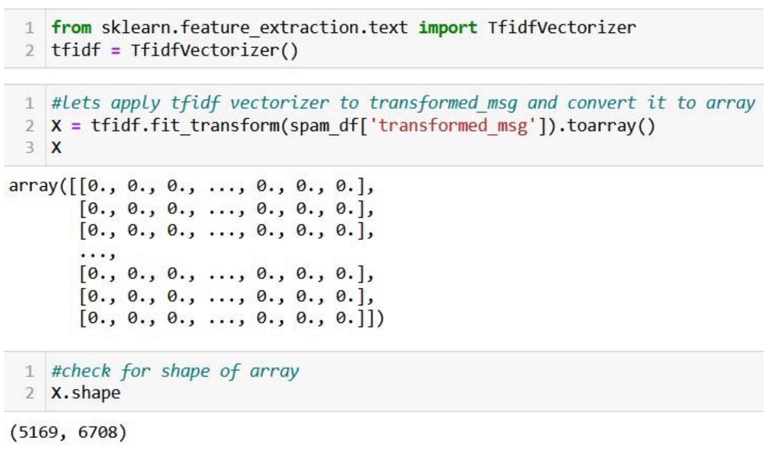




From these bar plots we noticed that the word ‘call’ occurs more than 300 times in spam messages and the word ‘u’ occurs near about 900 times in ham messages.

2.5 Model Building

Here we applied Tfidf vectorizer to the feature ‘transformed_msg’ and converted it into an array.



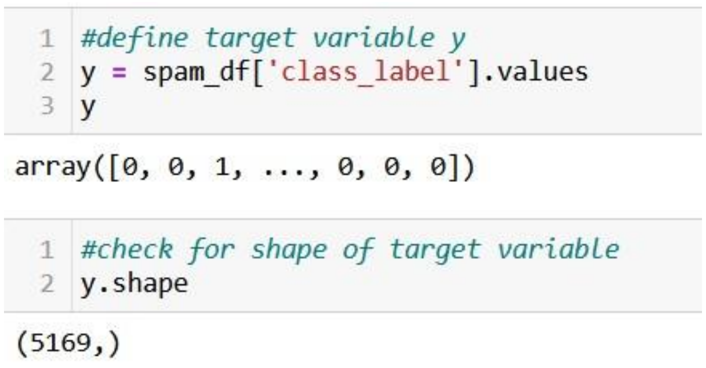
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
```

```
#lets apply tfidf vectorizer to transformed_msg and convert it to array
X = tfidf.fit_transform(spam_df['transformed_msg']).toarray()
X
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
#check for shape of array
X.shape
```

```
(5169, 6708)
```



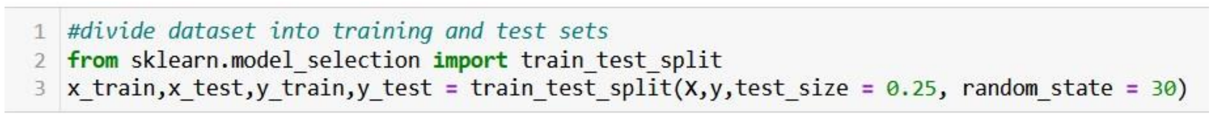
```
#define target variable y
y = spam_df['class_label'].values
y
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
#check for shape of target variable
y.shape
```

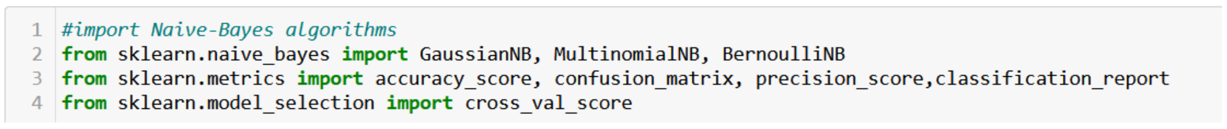
```
(5169,)
```

Then divided the dataset into training and test data using train-test-split technique. Here we took 25% of data as test data and remaining 75% as training data. And the random state is taken as 30.



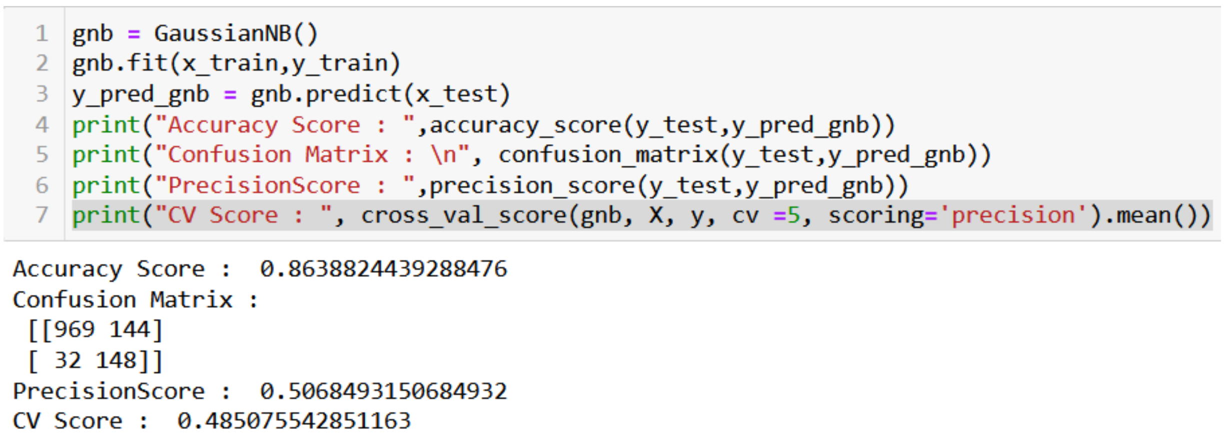
```
#divide dataset into training and test sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.25, random_state = 30)
```

The Naïve-Bayes algorithms are imported for further modelling.



```
#import Naive-Bayes algorithms
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,classification_report
from sklearn.model_selection import cross_val_score
```

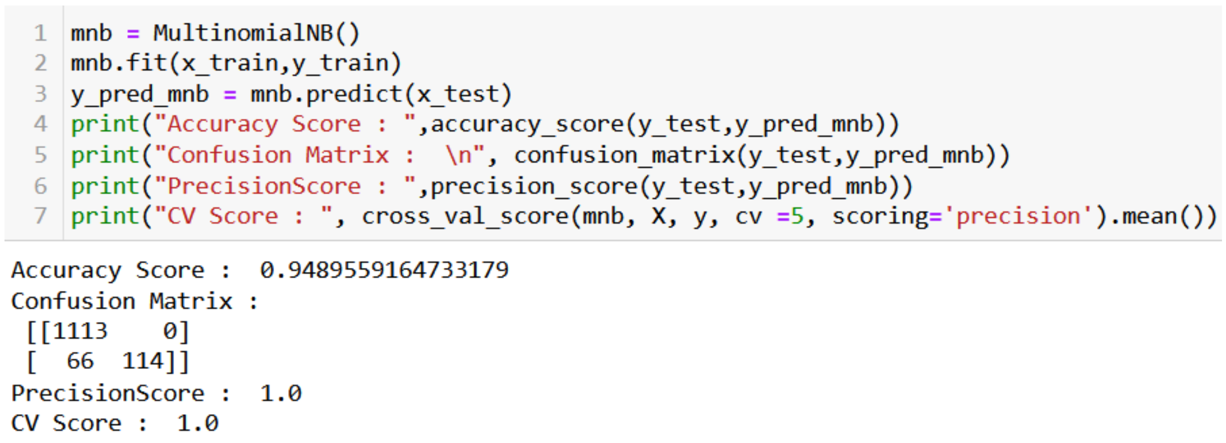
GaussianNB



```python
gnb = GaussianNB()
gnb.fit(x_train,y_train)
y_pred_gnb = gnb.predict(x_test)
print("Accuracy Score : ",accuracy_score(y_test,y_pred_gnb))
print("Confusion Matrix : \n", confusion_matrix(y_test,y_pred_gnb))
print("PrecisionScore : ",precision_score(y_test,y_pred_gnb))
print("CV Score : ", cross_val_score(gnb, X, y, cv =5, scoring='precision').mean())
```

```
Accuracy Score :  0.8638824439288476
Confusion Matrix :
 [[969 144]
 [ 32 148]]
PrecisionScore :  0.5068493150684932
CV Score :  0.485075542851163
```

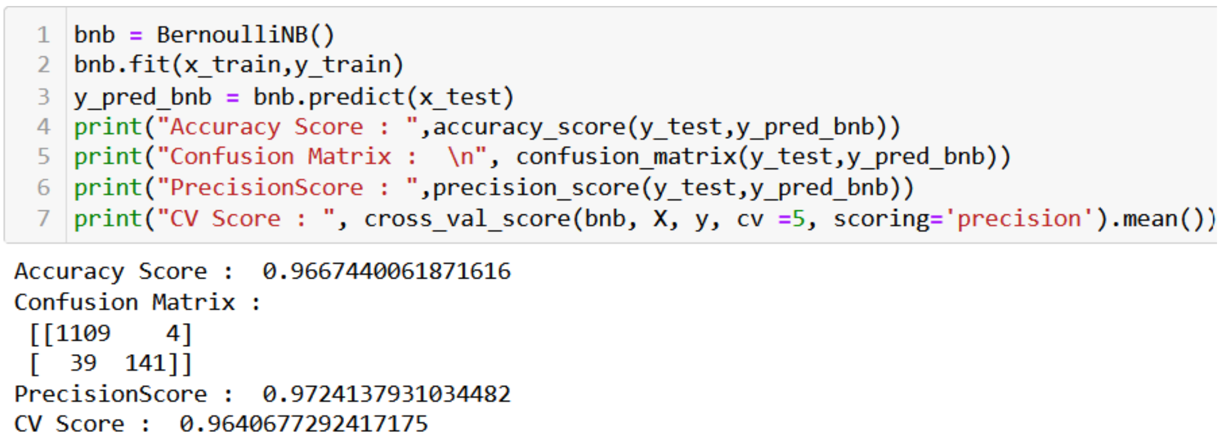
MultinomialNB



```python
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
y_pred_mnb = mnb.predict(x_test)
print("Accuracy Score : ",accuracy_score(y_test,y_pred_mnb))
print("Confusion Matrix :  \n", confusion_matrix(y_test,y_pred_mnb))
print("PrecisionScore : ",precision_score(y_test,y_pred_mnb))
print("CV Score : ", cross_val_score(mnb, X, y, cv =5, scoring='precision').mean())
```

```
Accuracy Score :  0.9489559164733179
Confusion Matrix :
 [[1113    0]
 [  66  114]]
PrecisionScore :  1.0
CV Score :  1.0
```

BernoulliNB

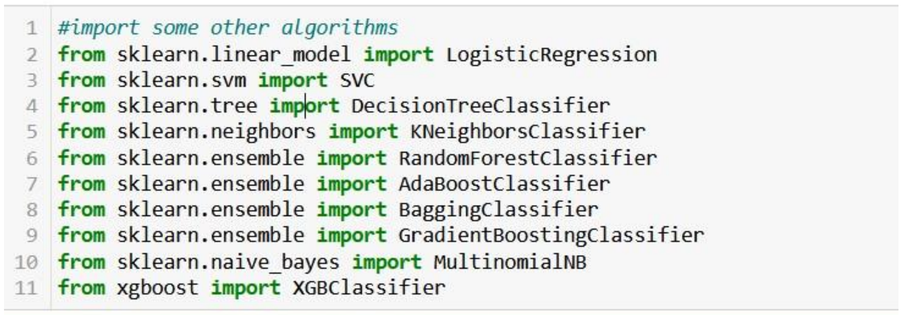


```python
bnb = BernoulliNB()
bnb.fit(x_train,y_train)
y_pred_bnb = bnb.predict(x_test)
print("Accuracy Score : ",accuracy_score(y_test,y_pred_bnb))
print("Confusion Matrix :  \n", confusion_matrix(y_test,y_pred_bnb))
print("PrecisionScore : ",precision_score(y_test,y_pred_bnb))
print("CV Score : ", cross_val_score(bnb, X, y, cv =5, scoring='precision').mean())
```

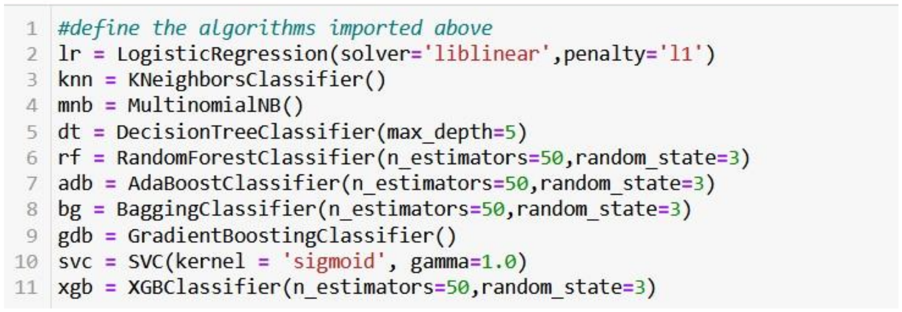
```
Accuracy Score :  0.9667440061871616
Confusion Matrix :
 [[1109    4]
 [  39  141]]
PrecisionScore :  0.9724137931034482
CV Score :  0.9640677292417175
```

Since, out of the 3 NB algorithms the MultinomialNB gives best precision score with great accuracy. Hence, we will select this one.

Then we imported some more algorithms and compared their accuracy scores and precision scores.

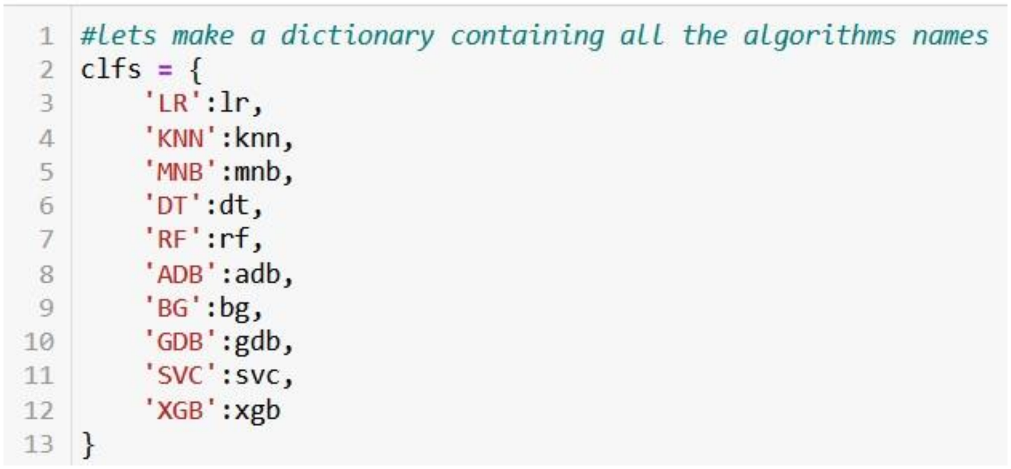


```python
#import some other algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
```



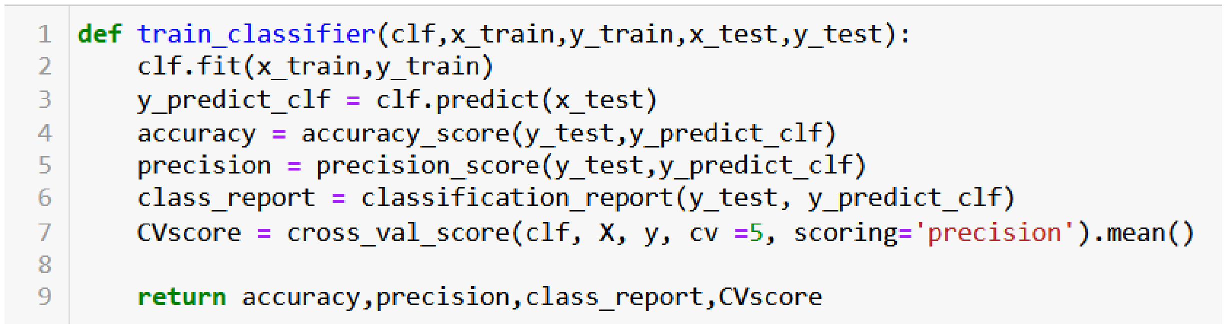
```python
#define the algorithms imported above
lr = LogisticRegression(solver='liblinear',penalty='l1')
knn = KNeighborsClassifier()
mnb = MultinomialNB()
dt = DecisionTreeClassifier(max_depth=5)
rf = RandomForestClassifier(n_estimators=50,random_state=3)
adb = AdaBoostClassifier(n_estimators=50,random_state=3)
bg = BaggingClassifier(n_estimators=50,random_state=3)
gdb = GradientBoostingClassifier()
svc = SVC(kernel = 'sigmoid', gamma=1.0)
xgb = XGBClassifier(n_estimators=50,random_state=3)
```

Then created a dictionary containing all the imported algorithms

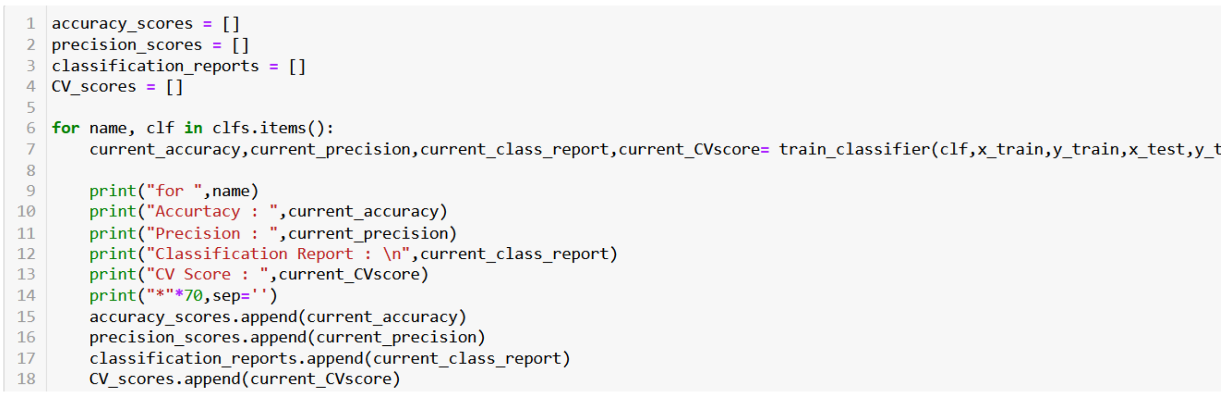


```python
#lets make a dictionary containing all the algorithms names
clfs = {
     'LR':lr,
     'KNN':knn,
     'MNB':mnb,
     'DT':dt,
     'RF':rf,
     'ADB':adb,
     'BG':bg,
     'GDB':gdb,
     'SVC':svc,
     'XGB':xgb
}
```

Further defined a function 'train_classifier' which will give us accuracy score and precision score for each of the algorithms.



```python
def train_classifier(clf,x_train,y_train,x_test,y_test):
    clf.fit(x_train,y_train)
    y_predict_clf = clf.predict(x_test)
    accuracy = accuracy_score(y_test,y_predict_clf)
    precision = precision_score(y_test,y_predict_clf)
    class_report = classification_report(y_test, y_predict_clf)
    CVscore = cross_val_score(clf, X, y, cv =5, scoring='precision').mean()

    return accuracy,precision,class_report,CVscore
```



```
accuracy_scores = []
precision_scores = []
classification_reports = []
CV_scores = []

for name, clf in clfs.items():
    current_accuracy,current_precision,current_class_report,current_CVscore= train_classifier(clf,x_train,y_train,x_test,y_t

    print("for ",name)
    print("Accurtacy : ",current_accuracy)
    print("Precision : ",current_precision)
    print("Classification Report : \n",current_class_report)
    print("CV Score : ",current_CVscore)
    print("*"*70,sep='')
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
    classification_reports.append(current_class_report)
    CV_scores.append(current_CVscore)
```

The following table shows the accuracy scores and precision scores of all implemented algorithms.

Sr no.	Algorithm	Accuracy score	Precision score	CV Score
1	KNN	0.894818	1.000000	1.000000
2	MNB	0.948956	1.000000	1.000000
3	RF	0.962877	1.000000	0.989473
4	SVC	0.965197	0.953020	0.963835
5	XGB	0.962104	0.939597	0.957739
6	LR	0.945862	0.936508	0.930299
7	GDB	0.953596	0.928571	0.947339
8	ADB	0.955143	0.917808	0.920298
9	BG	0.955143	0.858824	0.863729
10	DT	0.928074	0.822222	0.817915

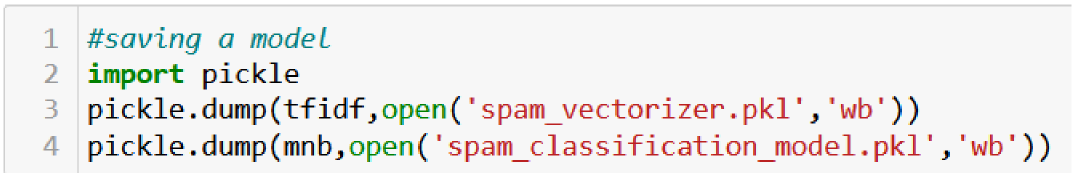
Now we can see that there are 3 algorithms giving best precision score as 1 with good accuracy scores.

Out of them we will select the Multinomial NB as our final algorithm which gives us precision score 1 and accuracy score 0.948956 with CV score 1.

Chapter 3

Model Saving

We saved the MultinomialNB as our final model

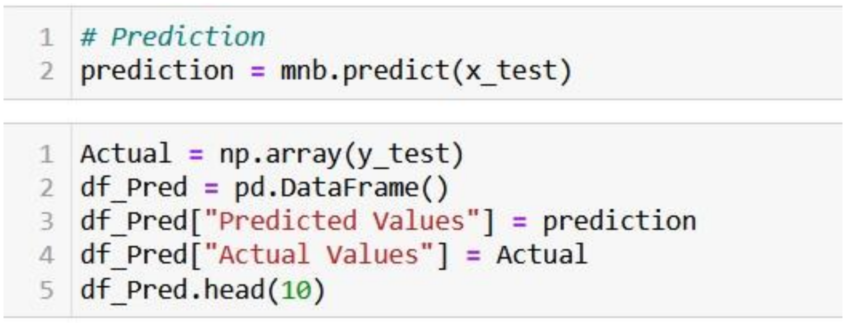


```python
#saving a model
import pickle
pickle.dump(tfidf,open('spam_vectorizer.pkl','wb'))
pickle.dump(mnb,open('spam_classification_model.pkl','wb'))
```

Result and Conclusion

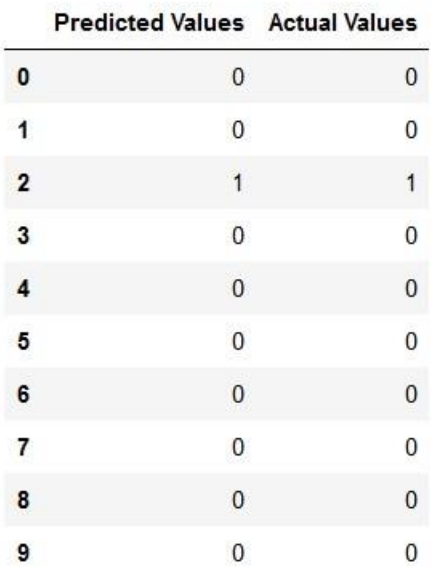
The algorithm selected as final model is MultinomialNB which gives us accuracy score 0.948956 and precision score 1 and CV score 0.989473

Test data Prediction



```python
# Prediction
prediction = mnb.predict(x_test)
```

```python
Actual = np.array(y_test)
df_Pred = pd.DataFrame()
df_Pred["Predicted Values"] = prediction
df_Pred["Actual Values"] = Actual
df_Pred.head(10)
```



	Predicted Values	Actual Values
0	0	0
1	0	0
2	1	1
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

Conclusion

In the study, we analysed machine learning techniques and their application to the field of spam filtering. A review of the algorithms been applied for classification of messages as either spam or ham is provided. The system architecture of email spam filter and the processes involved in filtering spam emails were looked into. The paper surveyed some of the publicly available datasets and performance metrics that can be used to measure the effectiveness and efficiency of any spam filter. The challenges of the machine learning algorithms in efficiently handling the menace of spam were pointed out and comparative studies of the machine learning techniques available in literature was done.

Scope and Limitations of this work Scope:

This project needs a coordinated scope of work. These scopes will help to focus on this project. The scopes are:

i. Modified existing machine learning algorithm.

ii. Make use and classify of a data set including data preparation, classification and visualization.

iii. Score of data to determine the accuracy of spam detection

iv. It provides sensitivity to the client and adapt well to the future spam techniques.

v. It considers a complete message instead of single word with respect to its organization.

vi. It increases security and control.

vii. It reduces IT administration cost and Network resource costs.

Limitations:

The limitations of this project are:

1. This project can only detect and calculate the accuracy of spam messages only.

2. It focuses on filtering, analysing and classifying the messages.

3. Do not block the messages.